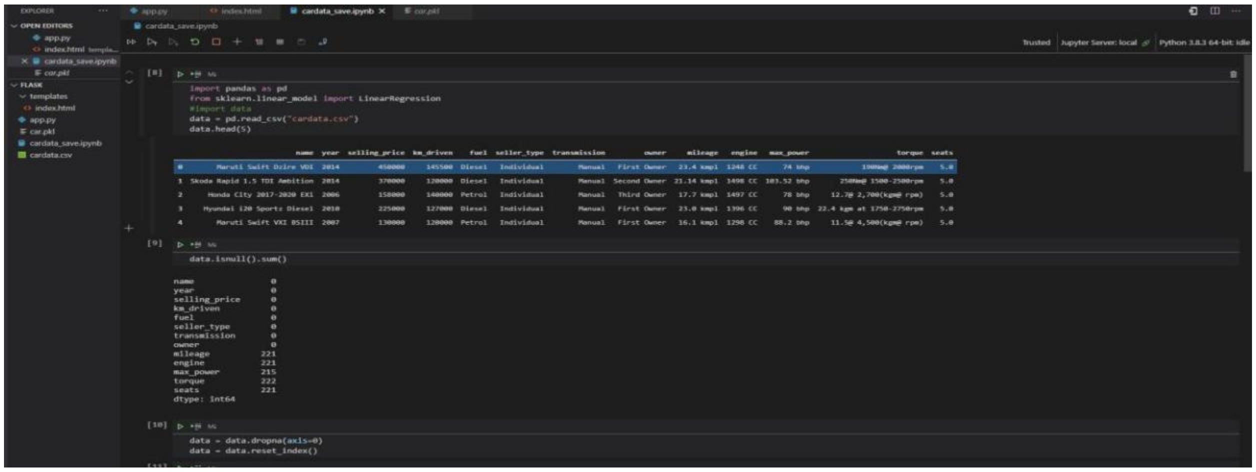
Deployment on Flask

Name: Osman Balli
Report date: 22.03.2021
Internship Batch: LISP01 - 13
Version:<1.0>
Proje located: https://github.com/osmanballi/Predict_Car_Price_with_Flask_and_Regression

Proposed Approach:

In this project, a web application was made that determines car prices according to their characteristics.



```
[8]
import pandas as pd
from sklearn.linear_model import LinearRegression
#import data
data = pd.read_csv("cardata.csv")
data.head(5)
```

```
                        name  year  selling_price  km_driven    fuel  seller_type  transmission         owner      mileage   engine   max_power                    torque  seats
0     Maruti Swift Dzire VDI  2014         450000     145500  Diesel   Individual        Manual   First Owner    23.4 kmpl  1248 CC      74 bhp            190Nm@ 2000rpm    5.0
1  Skoda Rapid 1.5 TDI Ambition  2014      370000     120000  Diesel   Individual        Manual  Second Owner   21.14 kmpl  1498 CC  103.52 bhp       250Nm@ 1500-2500rpm    5.0
2     Honda City 2017-2020 EXi  2006         158000     140000  Petrol   Individual        Manual   Third Owner    17.7 kmpl  1497 CC      78 bhp     12.7@ 2,700(kgm@ rpm)    5.0
3    Hyundai i20 Sportz Diesel  2010         225000     127000  Diesel   Individual        Manual   First Owner    23.0 kmpl  1396 CC      90 bhp  22.4 kgm at 1750-2750rpm    5.0
4       Maruti Swift VXI BSIII  2007         130000     120000  Petrol   Individual        Manual   First Owner    16.1 kmpl  1298 CC    88.2 bhp     11.5@ 4,500(kgm@ rpm)    5.0
```

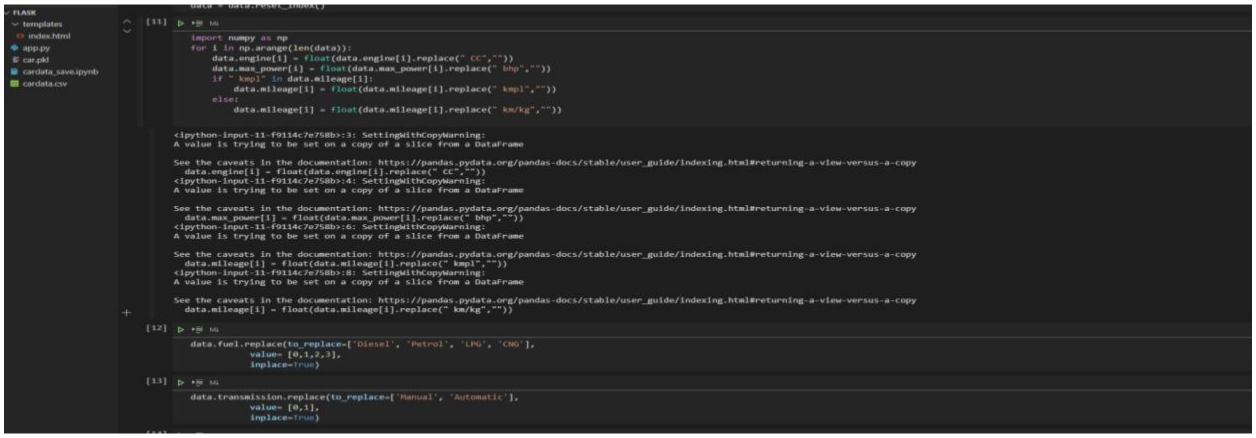
```
[9]
data.isnull().sum()
```

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```

```
[10]
data = data.dropna(axis=0)
data = data.reset_index()
```

Figure 1: Upload Data

The car data set consists of 13 columns. Here is information about Car Information, sales prices and features.



```
[11]
import numpy as np
for i in np.arange(len(data)):
    data.engine[i] = float(data.engine[i].replace(" CC",""))
    data.max_power[i] = float(data.max_power[i].replace(" bhp",""))
    if " kmpl" in data.mileage[i]:
        data.mileage[i] = float(data.mileage[i].replace(" kmpl",""))
    else:
        data.mileage[i] = float(data.mileage[i].replace(" km/kg",""))
```

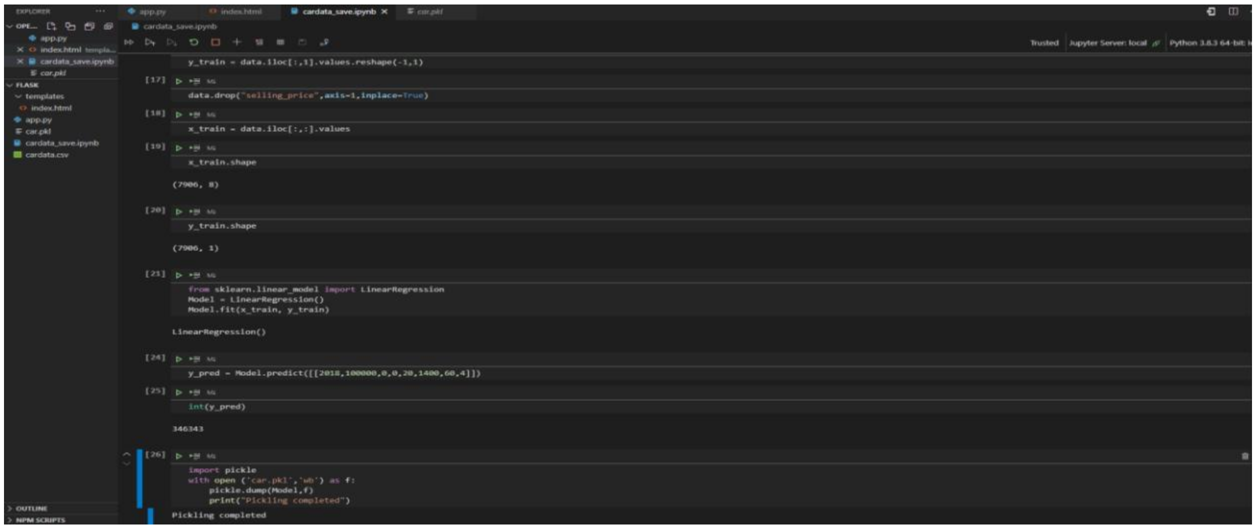
```
<ipython-input-11-f9114c7e758b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.engine[i] = float(data.engine[i].replace(" CC",""))
<ipython-input-11-f9114c7e758b>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.max_power[i] = float(data.max_power[i].replace(" bhp",""))
<ipython-input-11-f9114c7e758b>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.mileage[i] = float(data.mileage[i].replace(" kmpl",""))
<ipython-input-11-f9114c7e758b>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.mileage[i] = float(data.mileage[i].replace(" km/kg",""))
```

```
[12]
data.fuel.replace(to_replace=['Diesel', 'Petrol', 'LPG', 'CNG'],
            value= [0,1,2,3],
            inplace=True)
```

```
[13]
data.transmission.replace(to_replace=['Manual', 'Automatic'],
            value= [0,1],
            inplace=True)
```

Figure 2: Edit Data

Nan data was first detected and removed from the data set. It was also found that units were located next to some numerical data. These were corrected and converted into numerical data. Fuel and transmission were also converted to numerical data.



```
y_train = data.iloc[:,1].values.reshape(-1,1)
```

```
[17]
data.drop("selling_price",axis=1,inplace=True)
```

```
[18]
x_train = data.iloc[:,:].values
```

```
[19]
x_train.shape

(7906, 8)
```

```
[20]
y_train.shape

(7906, 1)
```

```
[21]
from sklearn.linear_model import LinearRegression
Model = LinearRegression()
Model.fit(x_train, y_train)

LinearRegression()
```

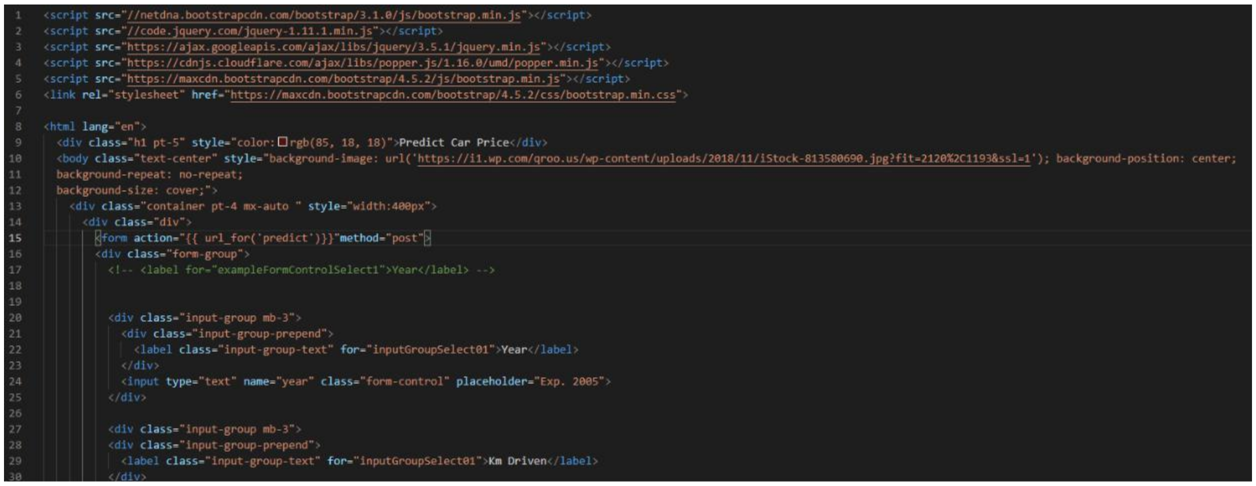
```
[24]
y_pred = Model.predict([[2018,100000,0,0,20,1400,60,4]])
```

```
[25]
int(y_pred)

346343
```

```
[26]
import pickle
with open ('car.pkl','wb') as f:
    pickle.dump(Model,f)
    print("Pickling completed")

Pickling completed
```

Figure 3: Training of data and recording of Model

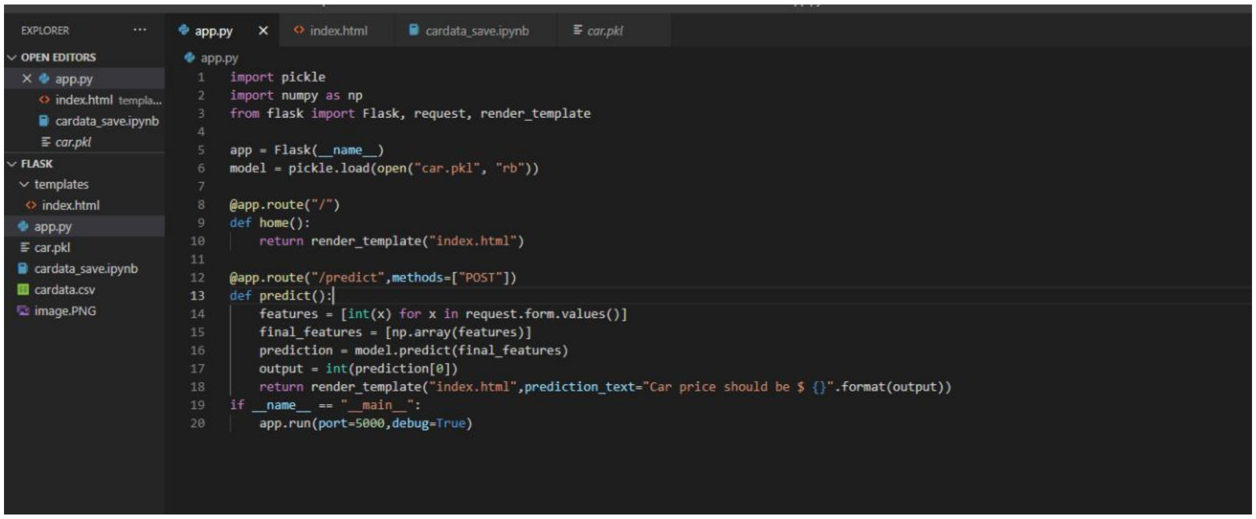
As the final stage of data-related operations, the data was trained and the model was recorded as car.pkl.



```
<script src="//netdna.bootstrapcdn.com/bootstrap/3.1.0/js/bootstrap.min.js"></script>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<html lang="en">
  <div class="h1 pt-5" style="color:rgb(85, 18, 18)">Predict Car Price</div>
  <body class="text-center" style="background-image: url('https://i1.wp.com/qroo.us/wp-content/uploads/2018/11/iStock-813580690.jpg?fit=2120%2C1193&ssl=1'); background-position: center;
  background-repeat: no-repeat;
  background-size: cover;">
    <div class="container pt-4 mx-auto " style="width:400px">
      <div class="div">
        <form action="{{ url_for('predict')}}"method="post">
        <div class="form-group">
          <!-- <label for="exampleFormControlSelect1">Year</label> -->


          <div class="input-group mb-3">
            <div class="input-group-prepend">
              <label class="input-group-text" for="inputGroupSelect01">Year</label>
            </div>
            <input type="text" name="year" class="form-control" placeholder="Exp. 2005">
          </div>

          <div class="input-group mb-3">
          <div class="input-group-prepend">
            <label class="input-group-text" for="inputGroupSelect01">Km Driven</label>
          </div>
```

Figure 4: Creating the index.html

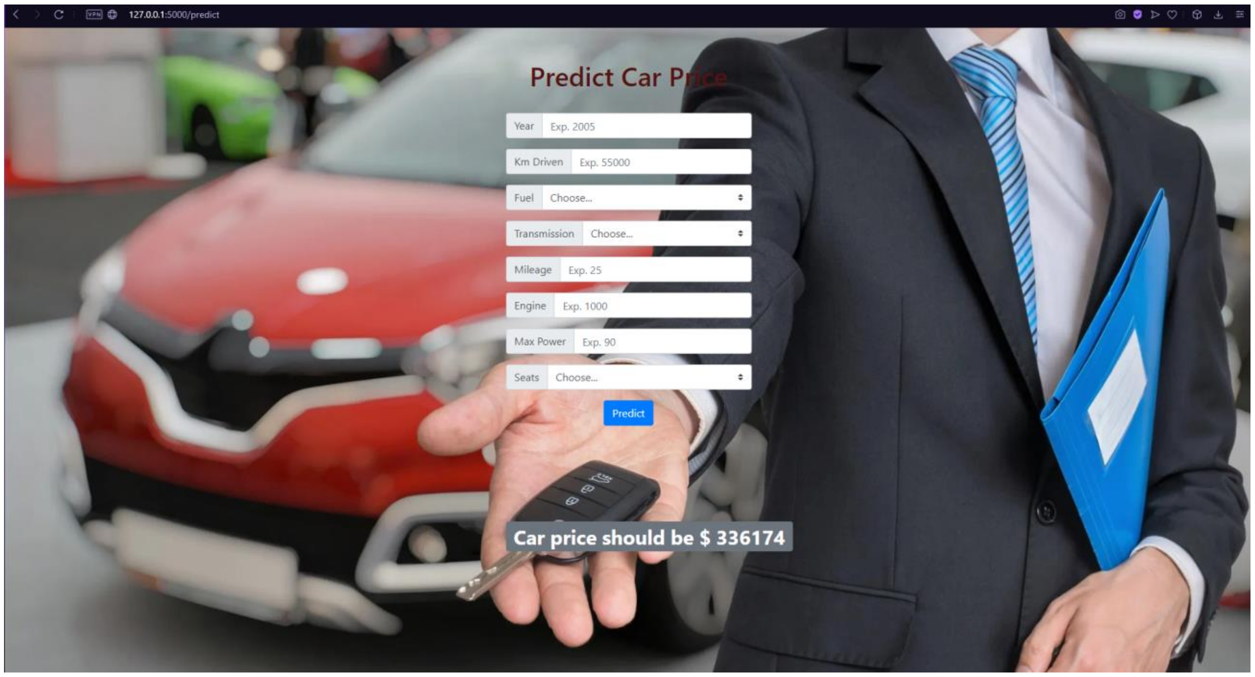
Bootsrap was used when creating the index.html. The Form was also created in the post method and the url was defined.



```python
import pickle
import numpy as np
from flask import Flask, request, render_template

app = Flask(__name__)
model = pickle.load(open("car.pkl", "rb"))

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict",methods=["POST"])
def predict():
    features = [int(x) for x in request.form.values()]
    final_features = [np.array(features)]
    prediction = model.predict(final_features)
    output = int(prediction[0])
    return render_template("index.html",prediction_text="Car price should be $ {}".format(output))
if __name__ == "__main__":
    app.run(port=5000,debug=True)
```

Figure 5: Main file

In this part, the model was loaded and estimated with data obtained from the post. The prediction was sent back to html.



Car price should be $ 336174

Figure 6: Launching the app on the browser